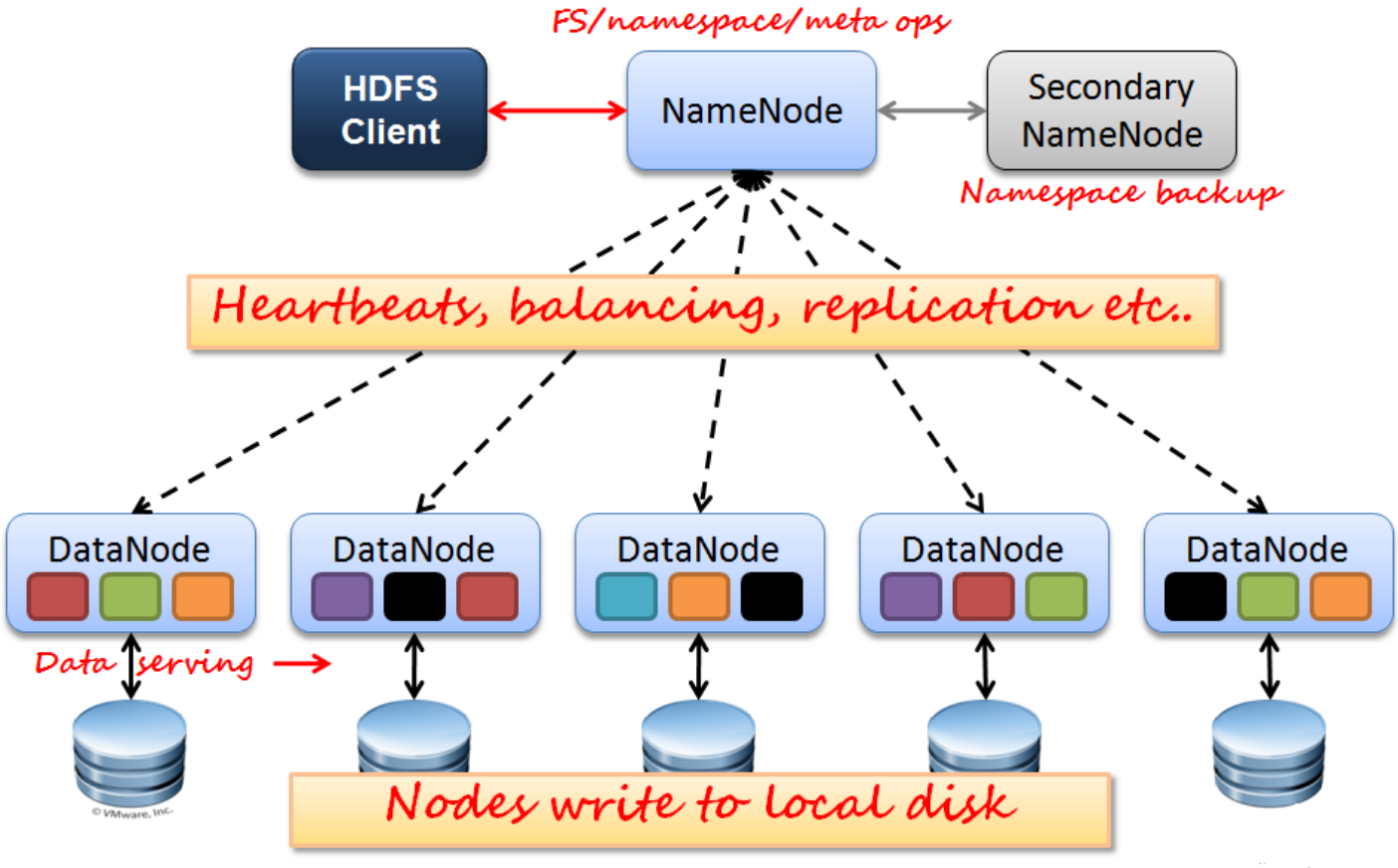
Jamzy Wang

Hadoop HDFS 详解

2015-07-24 21:49

HDFS 架构

HDFS是Hadoop应用中一个最主要的分布式存储系统。一个HDFS集群主要由一个 NameNode ,一个Secondary NameNode 和很多个 Datanode 组成：Namenode管理文件系统的元数据，而Datanode存储了实际的数据。客户端通过Namenode以获取文件的元数据或修饰属性，而真正的文件I/O操作是直接和Datanode进行交互的。



本文将详细介绍HDFS集群中的各个角色的作用以及工作原理、一些重要的特性。下面列出的是HDFS中常用特性的一部分：

- 机架感知（Rack awareness）：在调度任务和分配存储空间时考虑节点的物理位置。
- 安全模式：一种维护需要的管理模式。
- fsck：一个诊断文件系统健康状况的工具，能够发现丢失的文件或数据块。

- Rebalancer：当datanode之间数据不均衡时，平衡集群上的数据负载。
- 升级和回滚：在软件更新后有异常发生的情形下，能够回滚到HDFS升级之前的状态。
- Secondary Namenode：对文件系统名字空间执行周期性的检查点，将Namenode上HDFS改动日志文件的大小控制在某个特定的限度下。

HDFS优点：

(1) 适合大数据处理（支持GB，TB，PB级别的数据存储，支持百万规模以上的文件数量）

(2) 适合批处理（支持离线的批量数据处理，支持高吞吐率）

(3) 高容错性（以数据块存储，可以保存多个副本，容易实现负载均衡）

HDFS缺点：

(1) 小文件存取（占用namenode大量内存，浪费磁盘空间）

(2) 不支持并发写入（同一时刻只能有一个进程写入，不支持随机修改）

HDFS中的角色

文件系统的名字空间 (namespace)

HDFS支持传统的层次型文件组织结构。用户或者应用程序可以创建目录，然后将文件保存在这些目录里。文件系统名字空间的层次结构和大多数现有的文件系统类似：**用户可以创建、删除、移动或重命名文件。**HDFS暴露了文件系统的名字空间，用户能够以文件的形式在上面存储数据。

HDFS文件系统常见命令如下：

```
1  Command                        Descriptionhadoop
2  fs -mkdir mydir             Create a directory (mydir) in HDFShadoop
3  fs -ls                      List files and directories in HDFShadoop
4  fs -cat myfile              View a file contenthadoop
5  fs -du                      Check disk space usage in HDFShadoop
6  fs -expunge                 Empty trash on HDFShadoop
7  fs -chgrp hadoop file1      Change group membership of a filehadoop
8  fs -chown huser file1       Change file ownershiphadoop
9  fs -rm file1                Delete a file in HDFShadoop
10 fs -touchz file2            Create an empty filehadoop
11 fs -stat file1              Check the status of a filehadoop
12 fs -test -e file1           Check if file exists on HDFShadoop
13 fs -test -z file1           Check if file is empty on HDFShadoop
14 fs -test -d file1           Check if file1 is a directory on HDFS
```

用户存储下载文件命令如下：

```
1  Command                                                Description
2  hadoop fs -copyFromLocal <source> <destination>     Copy from local fileystem to HDFS
3  hadoop fs -copyFromLocal file1 data                 Copies file1 from local FS to data dir in HDFS
4  hadoop fs -copyToLocal <source> <destination>       copy from hdfs to local filesystem
5  hadoop fs -copyToLocal data/file1 /var/tmp          Copies file1 from HDFS data directory to /var/tmp on local FS
6  hadoop fs -put <source> <destination>               Copy from remote location to HDFS
7  hadoop fs -get <source> <destination>               Copy from HDFS to remote directory
8
9  hadoop distcp hdfs://192.168.0.8:8020/input hdfs://192.168.0.8:8020/output
10 Copy data from one cluster to another using the cluster URL
11
12 hadoop fs -mv file:///data/datafile /user/hduser/data
13 Move data file from the local directory to HDFShadoop
```

NameNode

HDFS采用master/slave架构。一个HDFS集群是由一个Namenode和一定数目的Datanodes组成。Namenode是一个中心服务器:

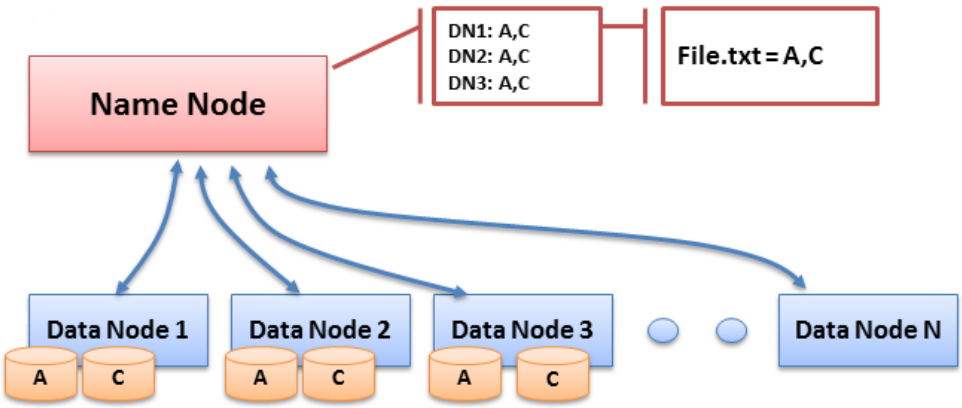
(1) 负责管理文件系统的名字空间操作，比如打开、关闭、重命名文件或目录。

(2) 负责确定数据块到具体Datanode节点的映射（文件到块的映射，块到DataNode的映射）。

(3) 监督Data nodes的健康

(4) 协调数据的存取

Datanode负责处理文件系统客户端的读写请求，在Namenode的统一调度下进行数据块的创建、删除和复制。



NameNode的特性可以总结如下：

(1) NameNode运行时所有数据都保存到内存，整个HDFS可存储的文件数受限于NameNode的内存大小。

(2) 一个block在NameNode中对应一条记录（一般一条记录占用150字节），如果是大量的小文件，会消耗大量内存。

(3) NameNode的数据会持久化到本地磁盘，但不保存block的位置信息，block由DataNode注册时上报和运行时维护

(4) NameNode失效则整个HDFS都失效了，所以要保证NameNode的可用性。

- **文件系统元数据的持久化**

Namenode上保存着HDFS的名字空间。对于任何对文件系统元数据产生修改的操作，Namenode都会使用一种称为EditLog的事务日志记录下来。例如，在HDFS中创建一个文件，Namenode就会在Editlog中插入一条记录来表示；同样地，修改文件的副本系数也将往Editlog插入一条记录。**Namenode在本地操作系统的文件系统中存储这个Editlog。整个文件系统的名字空间，包括数据块到文件的映射、文件的属性等，都存储在一个称为FsImage的文件中，这个文件也是放在Namenode所在的本地文件系统上。**

Namenode在内存中保存着整个文件系统的名字空间和文件数据块映射(Blockmap)的映像。这个关键的元数据结构设计得很紧凑，因而一个有4G内存的Namenode足够支撑大量的文件和目录。**当Namenode启动时，它从硬盘中读取Editlog和FsImage，将所有Editlog中的事务作用在内存中的FsImage上**，并将这个新版本的FsImage从内存中保存到本地磁盘上，然后删除旧的Editlog，因为这个旧的Editlog的事务都已经作用在FsImage上了。这个过程称为一个检查点(checkpoint)。在当前实现中，检查点只发生在Namenode启动时，在不久的将来将实现支持周期性的检查点。

- **元数据磁盘错误**

FsImage和Editlog是HDFS的核心数据结构。如果这些文件损坏了，整个HDFS实例都将失效。因而，Namenode可以配置成支持维护多个FsImage和Editlog的副本。任何对FsImage或者Editlog的修改，都将同步到它们的副本上。这种多副本的同步操作可能会降低Namenode每秒处理的名字空间事务数量。然而这个代价是可以接受的，因为即使HDFS的应用是数据密集的，它们也非元数据密集的。当Namenode重启的时候，它会选取最近的完整的FsImage和Editlog来使用。

- **单一NameNode的不足**

Namenode是HDFS集群中的单点故障(single point of failure)所在。如果Namenode机器故障，是需要手工干预的。目前，自动重启或在另一台机器上做Namenode故障转移的功能还没实现。

集群中单一Namenode的结构大大简化了系统的架构,但同时也带来了如下问题：

(1) NameNode的职责过重，无法避免单点故障

(2) 最大的瓶颈来自于内存，目前的NameNode，其元数据都是存储于单台服务器的内存里，那么其存储容量就受到了单台服务器内存容量的限制， 据估算，在装配100GB内存的服务器上，一般只能存储几亿级别的文件数。

(3) 随着集群规模的扩大，对于元数据的读写请求也会随之增多，那么元数据的访问性能也会受到单台服务器处理能力的限制。

那么如何对NameNode进行扩展呢？ 一种可行的方法是对NameNode的职责进行分离。

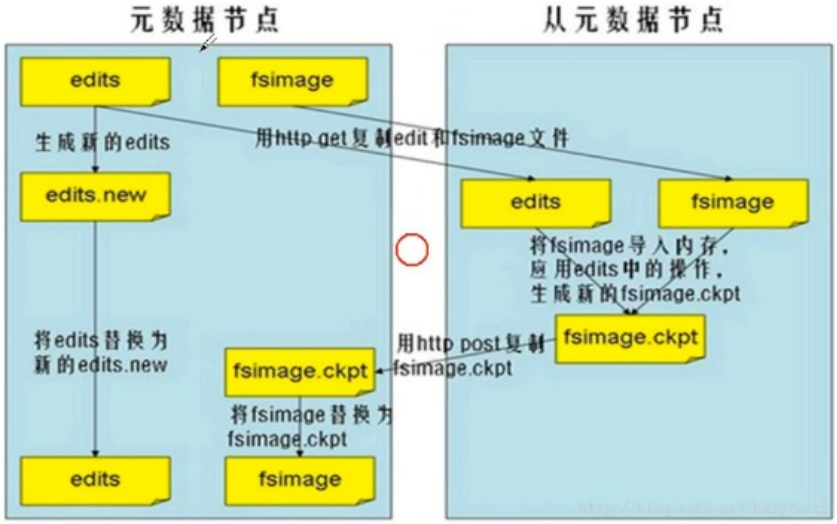
NameNode的职责可以分成两部分：1) 名字空间的管理; 2) 块管理：文件到块的映射，块到DataNode的映射。因此对其进行职责分离的解决方案可以为：NameNode只负责命名空间管理，在HDFS系统中新增加一个角色，这个角色专门负责块管理（文件到块的映射，块到DataNode的映射）

Secondary NameNode

NameNode将对文件系统的改动追加保存到本地文件系统上的一个日志文件（edits）。当一个NameNode启动时，它首先从一个映像文件（fsimage）中读取HDFS的状态，接着应用日志文件中的edits操作。然后它将新的HDFS状态写入（fsimage）中，并使用一个

空的edits文件开始正常操作。因为NameNode只有在启动阶段才合并fsimage和edits，所以久而久之日志文件可能会变得非常庞大，特别是对大型的集群。日志文件太大的另一个副作用是下一次NameNode启动会花很长时间。

Secondary NameNode定期(缺省为每小时)合并fsimage和edits日志，将edits日志文件大小控制在一个限度下。Secondary Namenode会连接到Namenode，同步Namenode的fsimage文件和edits文件。Secondary Namenode 合并fsimage文件和edits文件到一个新的文件中，保存到本地的同时把这些文件发送回NameNode。 当Namenode宕机，保存在Secondary Namenode中的文件可以用来恢复Namenode。在一个繁忙的集群中，系统管理员可以配置同步时间为更小的时间间隔，比如每分钟。



因为内存需求和NameNode在一个数量级上，所以**通常secondary NameNode和NameNode运行在不同的机器上。**Secondary NameNode通过bin/start-dfs.sh在conf/masters中指定的节点上启动。

总结一下Secondary Namenode： ＞ 1）不是NameNode的备份 2）周期性合并fsimage和editslog，并推送给NameNode 3）辅助恢复NameNode

DataNode

DataNode的特性可以总结为：

(1) 保存具体的block数据

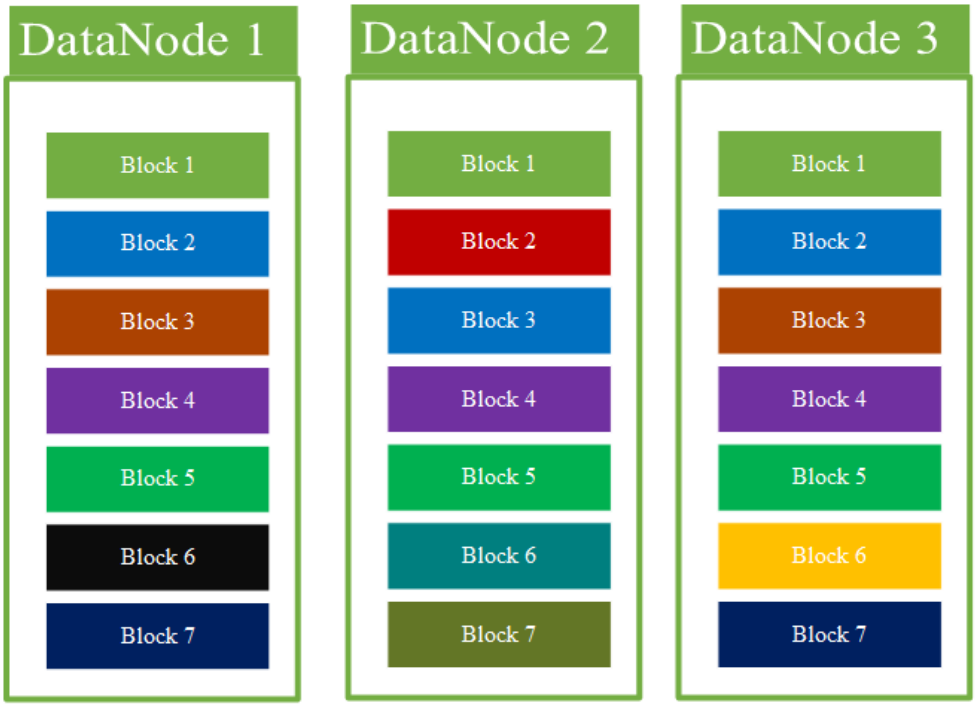
(2) 负责数据的读写操作和复制操作

(3) DataNode启动时会向NameNode报告当前存储的数据块信息，后续也会定时报告修改信息

(4) DataNode之间会进行通信，复制数据块，保证数据的冗余性

- **数据组织**

HDFS被设计成支持大文件，适用HDFS的是那些需要处理大规模的数据集的应用。这些应用都是**只写入数据一次，但却读取一次或多次，并且读取速度应能满足流式读取的需要。**HDFS支持文件的“一次写入多次读取”语义。一个大文件会被拆分成一个个的块(block)，然后存储于不同的DataNode上。如果一个文件小于一个block的大小，那么实际占用的空间为其文件的大小。



DataNode将HDFS数据以文件的形式存储在本地的文件系统中，它并不知道有关HDFS文件的信息。它把每个HDFS数据块（block）存储在本地文件系统的一个单独的文件中,每个块都会被复制到多台机器，默认复制3份。在DataNode中block是基本的存储单位（每次都是读写一个块），默认大小为64M。配置大的块主要是因为：

(1) 减少搜寻时间，一般硬盘传输速率比寻道时间要快，大的块可以减少寻道时间；

(2) 减少管理块的数据开销，每个块都需要在NameNode上有对应的记录；

(3) 对数据块进行读写，减少建立网络的连接成本

- **数据复制**

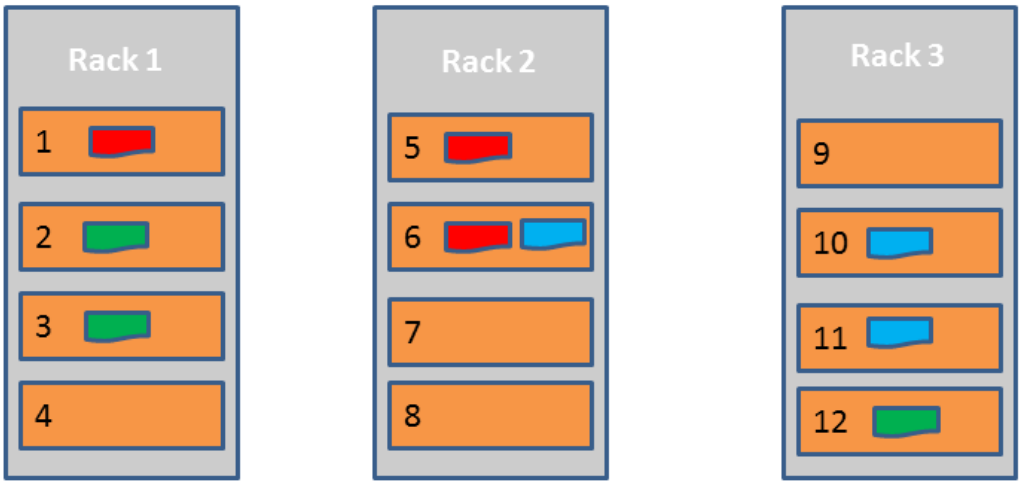
HDFS被设计成能够在一个大集群中跨机器**可靠**地存储超大文件。它将每个文件存储成一系列的数据块，除了最后一个，所有的数据块都是同样大小的。为了容错，文件的所有数据块都会有副本。每个文件的数据块大小和副本系数都是可配置的。应用程序可以指定某个文件的副本数目。副本系数可以在文件创建的时候指定，也可以在之后改变。**HDFS中的文件都是一次性写入的，并且严格要求在任何时候只能有一个写入者。**

Namenode全权管理数据块的复制，它周期性地从集群中的每个DataNode接收 心跳信号 和 块状态报告(Blockreport)。**接收到心跳信号意味着该DataNode节点工作正常。块状态报告包含了一个该Datanode上所有数据块的列表。**

副本的存放是HDFS可靠性和性能的关键。优化的副本存放策略是HDFS区分于其他大部分分布式文件系统的重要特性。这种特性需要做大量的调优，并需要经验的积累。HDFS采用一种称为 机架感知(rack-aware) 的策略来改进数据的可靠性、可用性和网络带宽的利用率。大型HDFS实例一般运行在跨越多个机架的计算机组成的集群上，不同机架上的两台机器之间的通讯需要经过交换机。在大多数情况下，同一个机架内的两台机器间的带宽会比不同机架的两台机器间的带宽大。

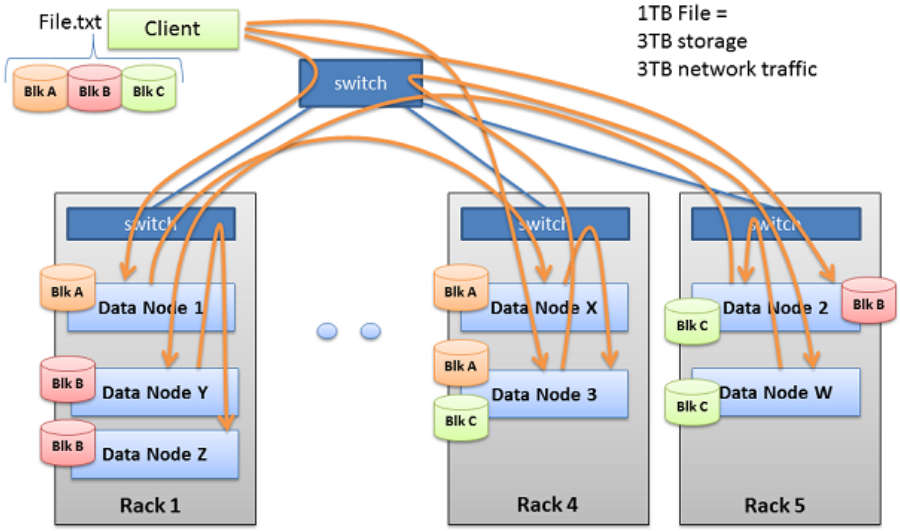
通过一个机架感知的过程，Namenode可以确定每个Datanode所属的机架id。一个简单但没有优化的策略就是将副本存放在不同的机架上。这样可以有效防止当整个机架失效时数据的丢失，并且允许读数据的时候充分利用多个机架的带宽。这种策略设置可以将副本均匀分布在集群中，有利于当组件失效情况下的负载均衡。但是，因为这种策略的一个写操作需要传输数据块到多个机架，这增加了写的代价。

在大多数情况下，副本系数是3，**HDFS的存放策略是将一个副本存放在本地机架的节点上，一个副本放在同一机架的另一个节点上，最后一个副本放在不同机架的节点上。**这种策略减少了机架间的数据传输，这就提高了写操作的效率。机架的错误远远比节点的错误少，所以这个策略不会影响到数据的可靠性和可用性。于此同时，因为数据块只放在两个（不是三个）不同的机架上，所以此策略减少了读取数据时需要的网络传输总带宽。在这种策略下，副本并不是均匀分布在不同的机架上。三分之一的副本在一个节点上，三分之二的副本在一个机架上，其他副本均匀分布在剩下的机架中，这一策略在不损害数据可靠性和读取性能的情况下改进了写的性能。

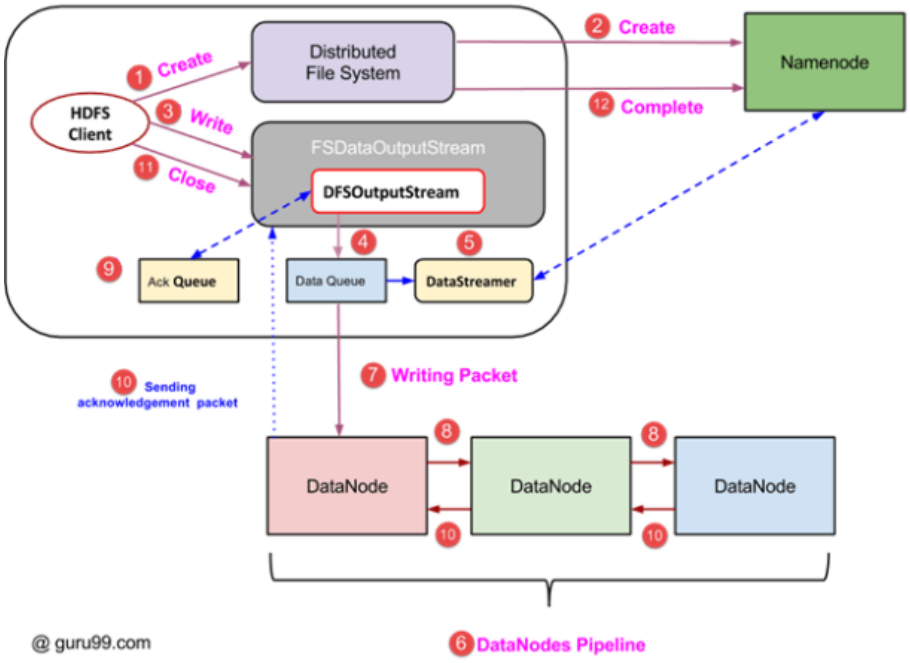


为了降低整体的带宽消耗和读取延时，HDFS会尽量让读取程序读取离它最近的副本。如果在读取程序的同一个机架上有一个副本，那么就读取该副本。如果一个HDFS集群跨越多个数据中心，那么客户端也将首先读本地数据中心的副本。

HDFS中的block数据3备份的复制采用的是的 流水线复制 方式,从前一个节点接收数据，并在同时转发给下一个节点，数据以流水线的方式从前一个DataNode复制到下一个。：



HDFS文件的写入具体过程如下图所示：



（1）客户端将文件写入本地磁盘的临时文件中（Staging）

（2）当临时文件大小达到一个block大小时，HDFS client通知NameNode申请写入文件

（3）NameNode在HDFS的文件系统中创建一个文件，并把该block id和要写入的DataNode的列表返回给客户端

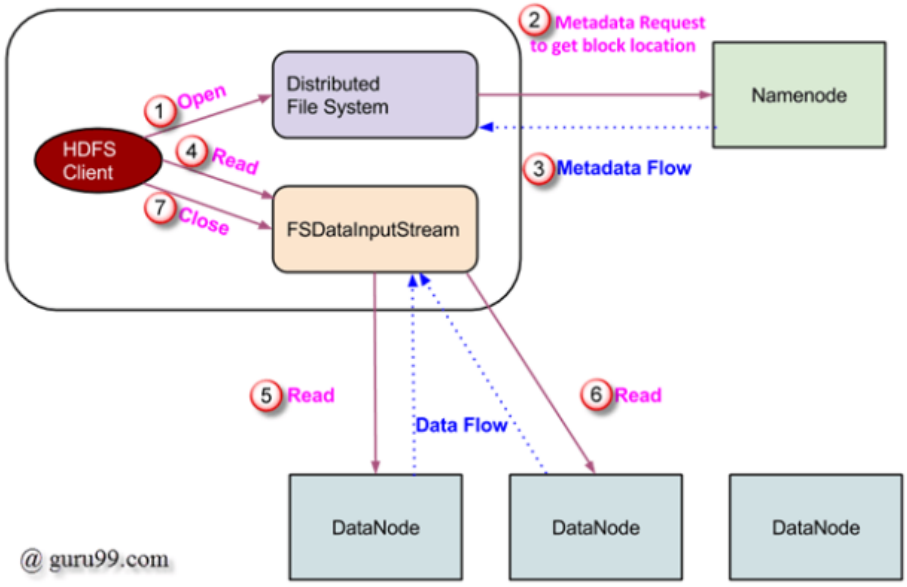
（4）客户端收到这些信息后，将临时文件写入DataNodes

```
1 （4.1） 客户端将文件内容写入第一个DataNode（一般以4kb为单位进行传输）
2 （4.2） 第一个DataNode接收后，将数据写入本地磁盘，同时也传输给第二个DataNode
3 （4.3） 依此类推到最后一个DataNode，数据在DataNode之间是通过pipeline的方式进行复制的
4 （4.4） 后面的DataNode接收完数据后，都会发送一个确认给前一个DataNode，最终第一个DataNode返回确认给客户端
5 （4.5） 当客户端接收到整个block的确认后，会向NameNode发送一个最终的确认信息
6 （4.6） 如果写入某个DataNode失败，数据会继续写入其他的DataNode。然后NameNode会找另外一个好的DataNode继续复制，以保证冗余性
7 （4.7） 每个block都会有一个校验码，并存放到独立的文件中，以便读的时候来验证其完整性
```

（5）文件写完后（客户端关闭），NameNode提交文件（这时文件才可见，如果提交前，NameNode垮掉，那文件也就丢失了。fsync：只保证数据的信息写到NameNode上，但并不保证数据已经被写到DataNode中）

当一个文件的副本系数被减小后，NameNode会选择过剩的副本删除。下次心跳检测时会将该信息传递给DataNode。Datanode遂即移除相应的数据块，集群中的空闲空间加大。

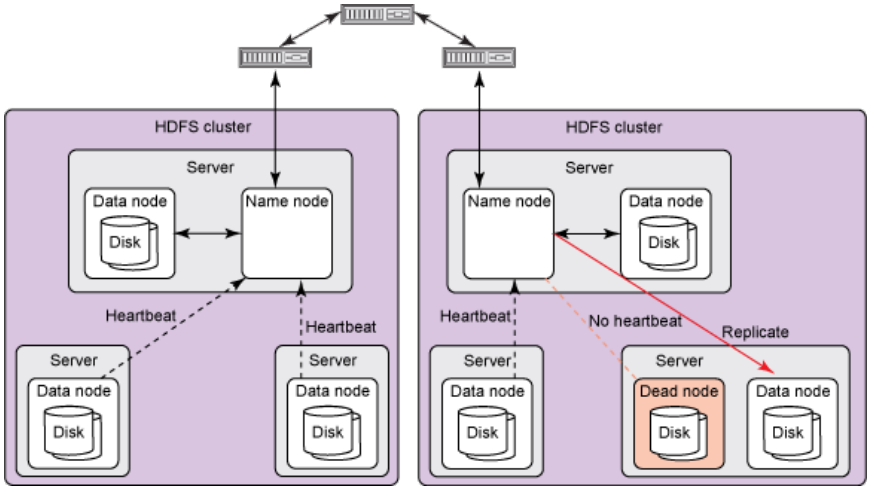
HDFS读文件流程：



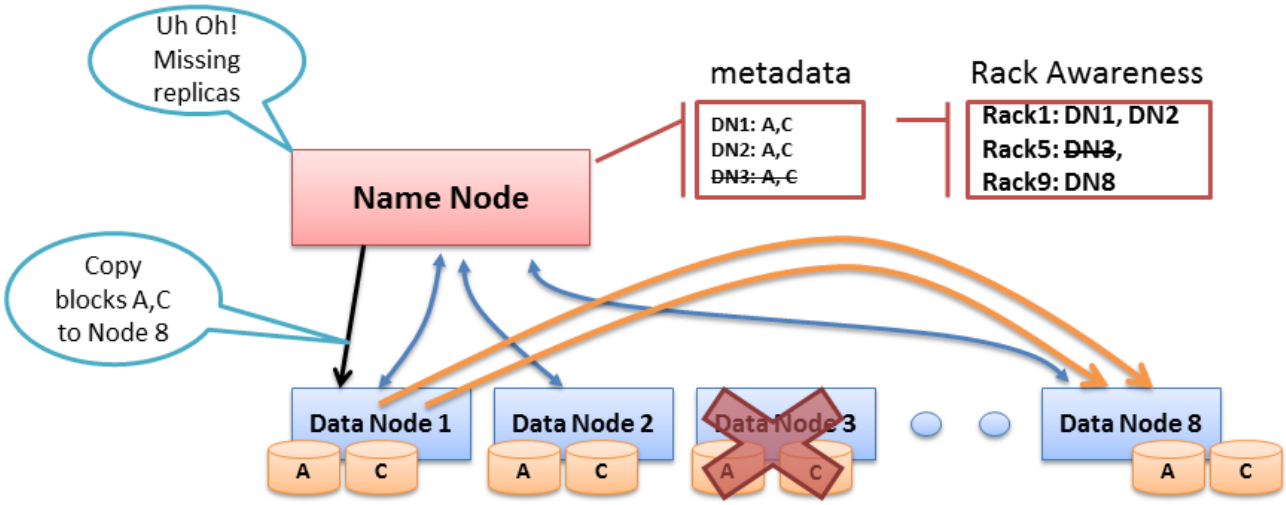
- 1.Hdfs 读数据的时候，客户端首先通过调用hdfs提供的open()读取目标文件，hdfs根据需要向NameNode节点查询这些文件存储在哪些DataNode上
- 2.NameNode返回所有文件block副本的DataNode节点的位置信息（并且根据他们与客户端的距离进行排序）
- 3.客户端从这些DataNode上读取block数据，一个block读完，继续寻找下一个block位置节点，直到读取完。
- 4.在读完一个block后，都会进行checksum校验，如果有错误，会通知NameNode，并且从下一个包含block的Datanode继续读。

- **数据的可靠性**

HDFS 的一个重要目标是可靠存储数据，即使在 NameNode、 DataNode或网络分区中出现故障。HDFS 克服故障的第一个步骤是探测。HDFS 使用心跳消息来探测 NameNode 和 DataNode 之间的连通性。HDFS 心跳有几种情况可能会导致 NameNode 和 DataNode 之间的连通性丧失。因此，每个 DataNode 都向它的 NameNode 发送定期心跳消息，这样，如果 NameNode 不能接收心跳消息，就表明连通性丧失。



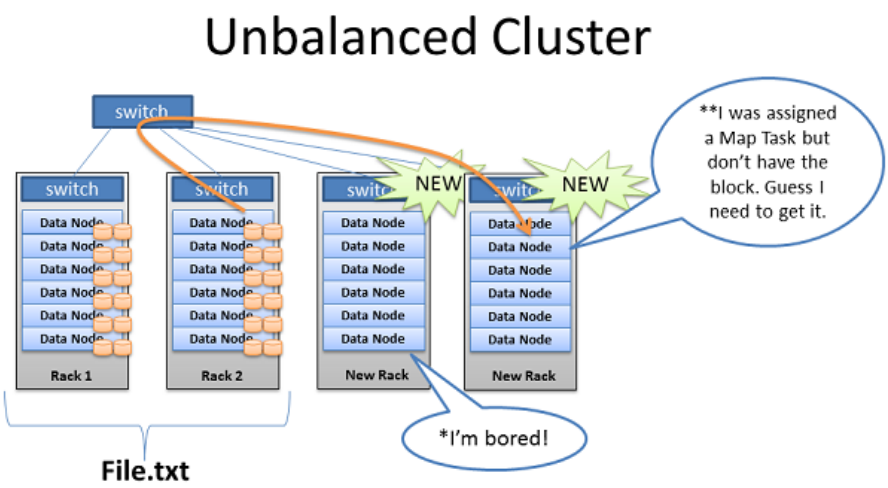
NameNode 将不能响应心跳消息的 DataNode 标记为 “死 DataNode ” ，并不再向它们发送请求。 存储在一个死节点上的数据不再对那个节点的 HDFS 客户端可用，该节点将被从系统有效地移除。如果一个节点的死亡导致数据块的复制因子降至最小值之下，Name node 将启动附加复制，将复制因子带回正常状态。



DataNode每3秒钟发送一个heartbeats给NameNode ，二者使用TCP9000端口的TCP握手来实现heartbeats。每十个heartbeats会有一个block report，Data node告知它所保存的数据块。block report使得Namenode能够重建它的metadata以确保每个数据block有足够的copy，并且分布在不同的机架上。

- **Rebalancer**

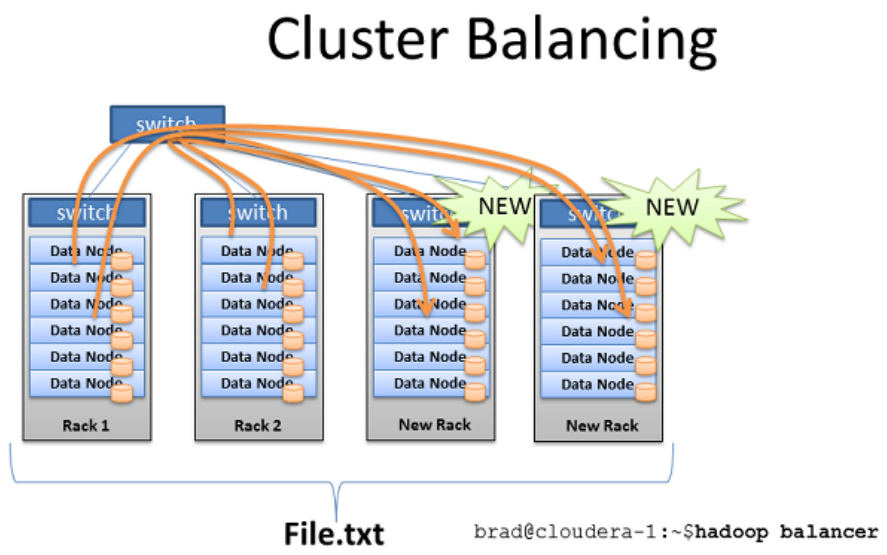
HDFS的数据也许并不是非常均匀的分布在各个DataNode中。一个常见的原因是在现有的集群上经常会增添新的DataNode节点。



当新增一个数据块（一个文件的数据被保存在一系列的块中）时，NameNode在选择DataNode接收这个数据块之前，会考虑到很多因素。其中的一些考虑的是：

(1) 将数据块的一个副本放在正在写这个数据块的节点上。 (2) 尽量将数据块的不同副本分布在不同的机架上，这样集群可在完全失去某一机架的情况下还能存活。 (3) 一个副本通常被放置在和写文件的节点同一机架的某个节点上，这样可以减少跨越机架的网络I/O。 (4) 尽量均匀地将HDFS数据分布在集群的DataNode中。

由于上述多种考虑需要取舍，数据可能并不会均匀分布在DataNode中。HDFS为提供了一个工具（balancer 命令），用于分析数据块分布和重新平衡DataNode上的数据分布。



HDFS重要特性

- **通讯协议**

所有的HDFS通讯协议都是建立在TCP/IP协议之上。客户端通过一个可配置的TCP端口连接到NameNode，通过ClientProtocol协议与NameNode交互。而DataNode使用DataNodeProtocol协议与Namenode交互。一个远程过程调用(RPC)模型被抽象出来封装ClientProtocol和Datanodeprotocol协议。在设计上，Namenode不会主动发起RPC，而是响应来自客户端或 Datanode 的RPC请求。

- **安全模式**

NameNode启动后会进入一个称为安全模式的特殊状态。**处于安全模式的Namenode是不会进行数据块的复制的。**NameNode从所有的 DataNode接收心跳信号和块状态报告。块状态报告包括了某个DataNode所有的数据块列表。每个数据块都有一个指定的最小副本数。当Namenode检测确认某个数据块的副本数目达到这个最小值，那么该数据块就会被认为是副本安全(safely replicated)的；在一定百分比（这个参数可配置）的数据块被NameNode检测确认是安全之后（加上一个额外的30秒等待时间），NameNode将退出安全模式状态。接下来它会确定还有哪些数据块的副本没有达到指定数目，并将这些数据块复制到其他Datanode上。

- **数据完整性**

从某个Datanode获取的数据块有可能是损坏的，损坏可能是由DataNode的存储设备错误、网络错误或者软件bug造成的。HDFS客户端软件实现了对HDFS文件内容的校验和(checksum)检查。当客户端创建一个新的HDFS文件，会计算这个文件每个数据块的校验和，

并将校验和作为一个单独的隐藏文件保存在同一个HDFS名字空间下。当客户端获取文件内容后，它会检验从DataNode获取的数据跟相应的校验和文件中的校验和是否匹配，如果不匹配，客户端可以选择从其他DataNode获取该数据块的副本。

- **存储空间回收**

当用户或应用程序删除某个文件时，这个文件并没有立刻从HDFS中删除。实际上，HDFS会将这个文件重命名转移到/trash目录。只要文件还在/trash目录中，该文件就可以被迅速地恢复。文件在/trash中保存的时间是可配置的，当超过这个时间时，Namenode就会将该文件从名字空间中删除。删除文件会使得该文件相关的数据块被释放。注意，从用户删除文件到HDFS空闲空间的增加之间会有一定时间的延迟。

Ref

- http://bradhedlund.com/2011/09/10/understanding-hadoop-clusters-and-the-network/
- http://simply-tutorial.com/blog/2013/11/01/learn-hdfs-a-beginners-guide/
- https://developer.yahoo.com/hadoop/tutorial/module2.html
- http://pennywong.gitbooks.io/hadoop-notebook/content/hdfs/introduction.html
- http://hadoop.apache.org/docs/r1.0.4/cn/hdfs_user_guide.html
- http://hadoop.apache.org/docs/r1.0.4/cn/hdfs_design.html
- http://pennywong.gitbooks.io/hadoop-notebook/content/hdfs/command.html

Posted by Jamzy Wang hadoop